REMARKS

Claims 1-20 are currently pending in the application. By this amendment, claims 5, 8, 9 and 20 are amended for the Examiner's consideration. The above amendments do not add new matter to the application and are fully supported by the specification. The specification and figures are also amended. Reconsideration of the rejected claims in view of the above amendments and the following remarks is respectfully requested.

Specification

Applicants amended the specification to correct a grammatical error.

Objection to Drawings

The drawings were objected to for being informal. Applicants traverse this objection.

Applicants submit that the figures satisfy the USPTO requirements according to 37 CFR 1.184 in that the figures can be reproduced with proper margins. In additional, Applicants note that the Examiner, on the coverpage of the instant office action, has indicated that the figures are accepted. Moreover, The Patent Office no longer requires formal drawings, per se. In any event, in an attempt to satisfy the Examiner, Applicants are attaching hereto revised Figures 1-3. The revised figures no longer include handwritten reference numerals.

35 U.S.C. §101 Rejection

Claim 20 was rejected under 35 U.S.C. §101. In an attempt to satisfy the Examiner, claim 20 is amended as follows:

> A computer program product comprising a computer usable medium having readable program code embodied in the medium operable to perform a method to:

Applicants submit that this language is equivalent to the language proposed by the Examiner. As such Applicants respectfully request that the rejection over claim 20 be withdrawn.

35 U.S.C. §102 Rejection

Claims 1-4, 9, 15, 16, 18 and 19 were rejected under 35 U.S.C. §102(e) for being anticipated by U. S. Patent Publication No. 2002/0095512 to Rana. This rejection is respectfully traversed.

Rejection of claims 9, 15, 16, 18 and 19

Applicants submit that the rejection of claims 9, 15, 16, 18 and 19 under 35 U.S.C. §102(e) is improper. Applicants note that claims 9, 15, 16, 18 and 19 have their dependencies originating from claim 8. As claim 8 is not rejected under 35 U.S.C. §102, using the Rana reference, Applicants submit that the rejection of claims 9, 15, 16, 18 and 19 is improper, since the subject matter of claim 8 is included in the claims 9, 15, 16, 18 and 19. Moreover, as the Examiner has admitted in the 35 U.S.C. §103(a) rejection that Rana does not include all of the features of claim 8, by definition, Rana cannot include all of the features of claims 9, 15, 16, 18 and 19, which depend from claim 8.

Accordingly, Applicants respectfully request that the rejection over claims 9, 15, 16, 18 and 19 be withdrawn.

Rejection of claim 1-4

Applicants submit that Rana does not show all of the features of claim 1-4, contrary to the Examiner's arguments. More specifically, the Examiner correctly notes that Rana shows a linked list. However, this linked list is not the same to that of the claimed invention.

By way of example, claim 1 recites, in pertinent part:

> comparing said context information of the received data packet to an expected sequence count for the given sequence, and storing the received packet with said context information in a memory as a linked list when there is a match, all received packets in the linked list being in order;
>
> creating a new linked list each time a new data packet is received out-of-order;
>
> linking in order all subsequent packets received in order to the new linked list;
>
> constructing a reorder table of addresses of the first packet for all linked lists; and
>
> reading packets out of the memory in an order specified by the reorder table.

By implementing the invention it is now possible to efficiently order packets received over a network. For example, the method detects breaks in sequences for one or more packet flows by detecting out-of-sequence packets and enters the segment of sequential packets into a separate memory area, such as a linked list, for a particular flow. Thus, in operation, for every flow, all packets from a single flow received in order are stored in memory as a linked list. In an embodiment, a linked list is used to store the pointer to the next packet in the sequence together with the packet data and packet status information. The last packet in the sequence terminates the linked list by placing this pointer to null, or some other terminator value. Paragraphs 0023 and 0024, reproduced below, show an example of the invention.

[0024] FIG. 2 is an illustrative diagram showing an embodiment of a received packet flow received out-of-order, generally denoted as 150. This exemplary out-of-order packet flow sequence is shown as 0, 1, 5, 2, 3, 4. In addition to the first received packet, the reorder table records those segments of packets (i.e., packet chains, where a chain includes one or more packets in order and may be just one packet) that are out-of-order from the last previously received packet. By way of example, as packets are received, the first packet with sequence number "0" is entered into the reorder table 120 at location 120a with the address of the associated packet address "addr1", 120b. The address, "addr.1", 120b, is the beginning of the linked-list, generally shown as 155, containing received in-sequence segment of packets 0 and 1.

[0025] Similarly, the next out-of-sequence packet is shown as 120c containing packet sequence 5 and can be found at memory location "addr.2", 120d, with the corresponding linked-list, creating a single packet chain, shown as reference numeral 160. Since the next packet following packet 5 is packet 2, packet 2 is considered out-of-sequence. Packet 2 is entered into the reorder table at 120e with "addr.3", 120f. Since the packets 3 and 4 follow in order after packet 2, they are linked into the linked-list following packet 2 creating a three deep packet chain, as shown generally as 165. Any further out-of-order segments may also have entries in the reorder table 120 and have associated linked-list chains similar to 165. Typically, every flow has an associated reorder table 120, expected sequence count 125, transmitted packet count 135, and associated linked-lists (e.g., 155, 160, 165, etc.). As sequences are entered into a reorder table, e.g., 120, an entry is also made into a transmission queue (also know as a transmit queue) such as 230 (FIG. 3).

However, Rana does not show these features. That is, the link list of Rana does not store the received packet with context information in a memory as a linked list when there is a match; that is, Rana does form the linked list when the received packets are in

an order. Rana also does not show creating a new linked list each time a new data packet is received out-of-order and linking in order all subsequent packets received in order to the new linked list. Instead, Rana shows providing a single linked list for all packets, without reference to whether there is an in order sequence.

In Rana, a Queue engine 10 takes network traffic in the form of data packets, (datagrams, or PDUs when unfragmented), from input 12 on a POS-PHY level 3 or equivalent data bus 44 and acts to buffer, reorder, and reassemble the datagrams. As noted in paragraph [0020], PDU assembler, or packet assembler, 26 is at the core of queue engine 10 and performs several functions. For example, after PDU assembler 26 receives the data packets from input interface it allocates blocks in link list memory 24 using link list memory controller 22. The link lists in link list memory 24 are used by queue engine 10 to track pointers associated with data packets stored in packet memory 20. PDU assembler 26 also deallocates blocks for error conditions including out of memory cases. The link list is thus used for out of order packets.

As further disclosed in Rana, if the PDU is not the next expected packet, it is processed as an out of order packet by IP reordering unit 34. IP reordering unit 34 uses IRU memory 44 to keep track of windows which reflect PDUs belonging to a particular traffic flow and where each PDU belongs in sequence. As with fragment reassembly unit 28, IP reordering unit 34 is able to modify the link lists in link list memory 24 using link list memory controller 22 in order to place PDUs in the correct order. Once the next expected PDU is received and placed into the proper place in sequence it is sent to link list control unit 40 for forwarding to output 42. As this specifically discloses, the link list is created from out of order packets. And, as discussed at paragraph [0030], link list control unit 40 accepts complete, in order PDU from PDU assembler 26, fragment reassembly unit 28 or IP reordering unit 34 into a queue for transmission to output 42.

Thus, as is clearly discernable from the above, Rana discloses only a single link list. Also, Rana does not disclose forming the linked list when the received packets are in an order. Rana also does not show creating a new linked list each time a new data

packet is received out-of-order and linking in order all subsequent packets received in order to the new linked list.

Accordingly, Applicants respectfully request that the rejection over claims 1-4, 9, 15, 16, 18 and 19 be withdrawn.

35 U.S.C. §103 Rejection

Claims 5-8, 10-14, 17 and 20 were rejected under 35 U.S.C. §103(a) for being unpatentable over Rana in view of U.S. Patent No. 6,381,242 to Maher, III. This rejection is respectfully traversed.

Claims 5-7

Applicants submit that claims 5-7 depend from an allowable base claim. As such, claims 5-7 include the features of the base claim. Accordingly, Applicants respectfully submit that claims 5-7 include allowable subject matter.

Claims 8, 10-14, 17 and 20

Independent claims 8 and 20 recite creating a linked list each time a new data packet is received out-of-sequence and linking in order all subsequent packets received in sequence to the linked list. As discussed above, Rana does not show or suggest these features. Additionally, Applicants submit that Maher also does not show or suggest these features and, as such, the combination of Rana and Maher do not teach or suggest the claimed invention.

In Maher, a content processor 110 must be able to reassemble fragmented packets and reorder out of order packets on a per session basis. Reordering and reassembling is the function of queue engine 302. Queue engine 302 receives data off the fast-path data bus 126 using fast-path interface 310. Packets are then sent to packet reorder and reassembly engine 312, which uses packet memory controller 316 to store the packets into packet memory 112. Reordering and reassembly engine 312

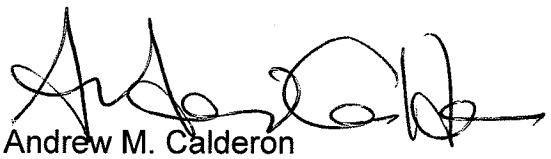
also uses link list controller 314 and link list memory 318 to develop detailed link lists that are used to order the data packets for processing. The data packets are broken into 256 byte blocks for storage within the queue engine 302. However, there is simply no suggestion that the link list is used to create a linked list each time a new data packet is received out-of-sequence and linking in order all subsequent packets received in sequence to the linked list.

Accordingly, Applicants respectfully request that the rejection over claims 8, 10-14, 17 and 20 be withdrawn.

CONCLUSION

In view of the foregoing amendments and remarks, Applicants submit that all of the claims are patentably distinct from the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue. The Examiner is invited to contact the undersigned at the telephone number listed below, if needed. Applicant hereby makes a written conditional petition for extension of time, if required. Please charge any deficiencies in fees and credit any overpayment of fees to Attorney's Deposit Account No. 09-0456.

Respectfully submitted,



Andrew M. Calderon
Registration No. 38,093

Greenblum & Bernstein, P.L.C.
1950 Roland Clarke Place
Reston, Virginia 20191
Telephone: 703-716-1191
Facsimile: 703-716-1180